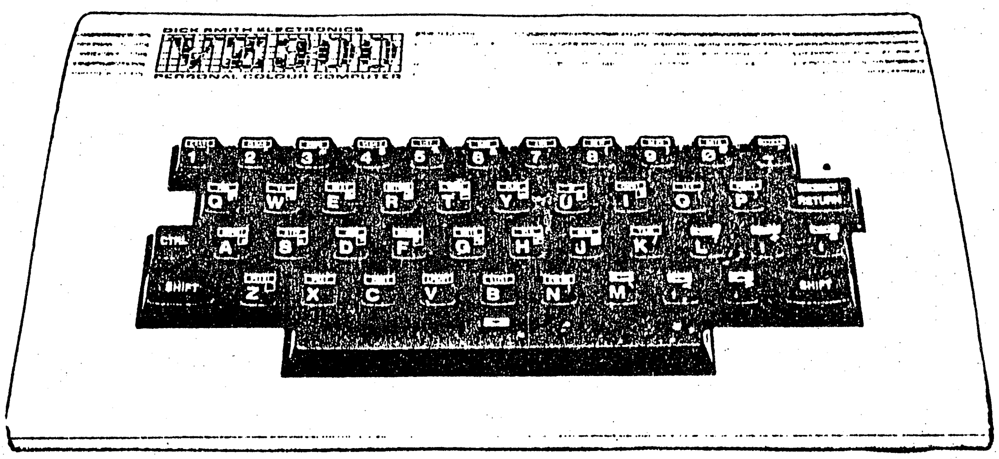
JAN./FEB. 1987 ISSUE # 8/9

# HUNTER VALLEY

# VZ JOURNAL



PRODUCED BY H.V.V.Z.U.G. . .

# COMMITTEE MEMBERS - - - - - - -

PETER ELLIS . . . . . . PRESIDENT

3 GOW STREET
HAMILTON NORTH
N.S.W. 2303
(049) 69 5697

MARK O'BRIEN . . . . . VICE PRESIDENT

46 FERN STREET
ISLINGTON
N.S.W. 2296
(049) 61 5490

ROSS WOODS . . . . . . SECRETARY

83 LAMBTON PARADE
SWANSEA HEADS
N.S.W. 2281
(049) 71 2843

LEIGH ROGERS . . . . . TREASURER

40 FLEET STREET
NEW LAMBTON
N.S.W. 2305
(049) 57 5738

JOE LEON . . . . . . . EDITOR

22 DRURY STREET
WALLSEND
N.S.W. 2287
(049) 51 2756

MAIL ALL SUBMISSIONS AND FUNDS TO :-

HUNTER VALLEY VZ USERS' GROUP
C/- P.O. BOX 161 JESMOND
N.S.W. 2299

First I would like to thank Dave Boyce, Dave Mitchell and Bob Kitch who submitted their articles on tape and disk. It greatly reduces the risk of errors creeping in. Thanks fellas. Some promised articles had to be held over till next issue.

SNOOPY CALENDAR PART 1 by Dave Boyce :- Page 4-6

Due to the length of this program part two will be in the next issue. Snoopy calendar consists of two parts. Part one is the calendar while part two is snoopy himself.

UTILITY REVIEW - W.P. TAPE/DISK :- Page 7

Tape W.Processor to Disk conversion utilities which will allow you to convert your Tape W.P. for full disk use and will also transfer your tape W.P. files to disk.

MERGING W. PROCESSOR FILES (TAPE OR DISK) :- Page 7

A brief description on the procedure. The insruction book only mentions that it's possible using the LOAD command.

INTRODUCTION TO PROGRAMMING BY B. KITCH :- Pages 8-10

This is the second informative part and contains a lot of good advice on how to become a competent programmer. Unfotunately a lot of people skip over many articles in preference to more attractive material in magazines. As Bob mentions it's very important to plan your program if you want to improve your skills.

OTHER VZ PUBLICATIONS :- Page 10

Ther are two other VZ publications besides ours. VZ USER is on the advanced side while LE'VZ 200/300 OOP is more general in application. Both are informative and exellent value for money.

TECHNICAL REPORT 1 by Joe Leon :- Pages 11-12

The project shows how to fit a write protect override switch, change the dull motor on led to a super bright one and a power on/off switch to your disk drive.

RESTORE BY D. MITCHELL :- Pages 13-15

This is a handy utility that will RESTORE your progam after using the NEW command. May also work after using reset button.

HIGH GAMES SCORE :- Page 15

Not much activity in this area but should see some new champs after the school holidays.

TECHNICAL REPORT 2 by Joe Leon :- Pages 16-17

Two circuits are shown for an electronic shift lock key.

UNDERSTANDING YOUR VZ by Robert Quinn :- Pages 18-20

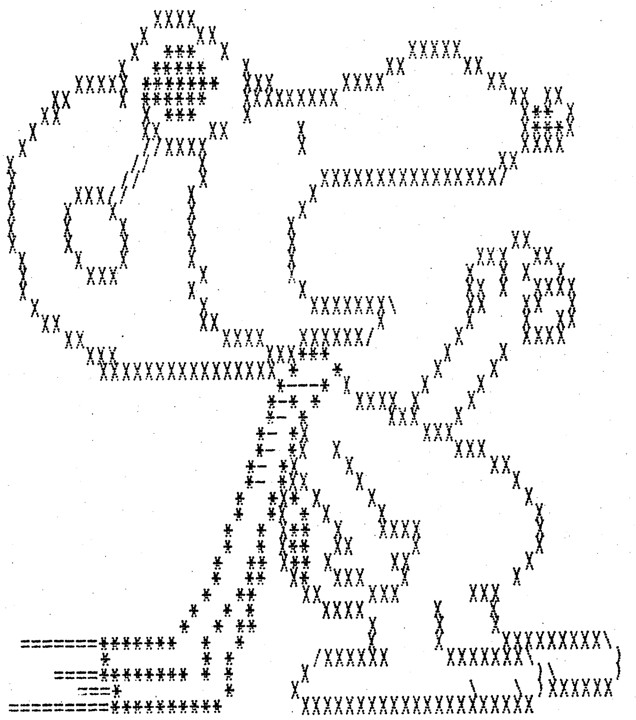
This is the final in the series and hopefully not the last we hear from Robert. Quite a few people should understand their VZ'S a lot better now. Much appreciated Robert.

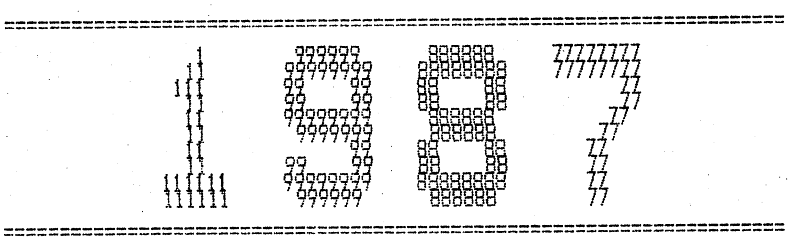
BYTE BACK :-

IT appears our members have more bark than byte as nobody bit back as yet. This section is for your comments or queries.

BELIEVE IT OR NOT :-

A 15 year old got a COMMODORE for Christmas, connected it all up, turned on the power and a READY message appeared on the screen. So he typed in YES and the latest news is he's still waiting. Things like that dont happen with the VZ, or do they ?






### JANUARY

| SU | MO | TU | WE | TH | FR | SA |
|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 |

### FEBRUARY

| SU | MO | TU | WE | TH | FR | SA |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 |

### MARCH

| SU | MO | TU | WE | TH | FR | SA |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 29 | 30 | 31 | | | | |

### APRIL

| SU | MO | TU | WE | TH | FR | SA |
|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 26 | 27 | 28 | 29 | 30 | | |

### MAY

| SU | MO | TU | WE | TH | FR | SA |
|---|---|---|---|---|---|---|
| 31 | | | | | 1 | 2 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 |

### JUNE

| SU | MO | TU | WE | TH | FR | SA |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | | | | |

### JULY

| SU | MO | TU | WE | TH | FR | SA |
|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 26 | 27 | 28 | 29 | 30 | 31 | |

### AUGUST

| SU | MO | TU | WE | TH | FR | SA |
|---|---|---|---|---|---|---|
| 30 | 31 | | | | | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 23 | 24 | 25 | 26 | 27 | 28 | 29 |

### SEPTEMBER

| SU | MO | TU | WE | TH | FR | SA |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29 | 30 | | | |

### OCTOBER

| SU | MO | TU | WE | TH | FR | SA |
|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 |

### NOVEMBER

| SU | MO | TU | WE | TH | FR | SA |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 29 | 30 | | | | | |

### DECEMBER

| SU | MO | TU | WE | TH | FR | SA |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29 | 30 | 31 | | |

```
90 '  SNOOPY CALENDAR FOR THE
100 ' BROTHER M-1009 OR EQUIV.
110 ' DOT MATRIX PRINTER.
120 ' FILE - SNOOPYCL
130 CLS:PRINT:PRINT"DO YOU HAVE YOUR PRINTER"
135 INPUT" TURNED ON <Y/N>";DF$
140 IF DF$="Y" THEN 155
145 PRINT:PRINT" OH WELL !! - SWITCH OFF AND    COME BACK WHEN";
150 PRINT" YOU HAVE - BYE!!":END
155 CLEAR800
162 VZ$="===================================="
165 DIM L(11),YR(11,6,4),X$(9,9)
170 DATA 31,28,31,30,31,30,31,31,30,31,30,31
172 '---X--------X-X--------X-X--------X-X--------X-X--------X
175 DATA" 000000 ","00000000","00    00","00    00","00    00"
180 DATA"00    00","00    00","00    00","00000000"," 000000 "
185 DATA"   1    ","   11   ","  111   ","   11   ","   11   "
190 DATA"  11    ","   11   ","   11   "," 111111 "," 111111 "
195 DATA" 222222 ","22222222","22    22","      22","  222222"
200 DATA" 222222 ","222     ","22      ","22222222","22222222"
205 DATA"33333333","33333333","     33 ","    33  ","   333  "
210 DATA"    333 ","     333","33    33","33333333"," 333333 "
215 DATA"     4  ","    44  ","   444  ","  4444  "," 44 44  "
220 DATA"44  44  ","44444444","44444444","    44  ","    44  "
225 DATA"55555555","55555555","55      ","55      ","5555555 "
230 DATA"55555555","      55","55    55","55555555"," 555555 "
235 DATA" 666666 ","66666666","66    66","66      ","6666666 "
240 DATA"66666666","66    66","66    66","66666666"," 666666 "
245 DATA"77777777","77777777","      77","      77","     77 "
250 DATA"    77  ","   77   ","   77   ","   77   ","   77   "
255 DATA" 888888 ","88888888","88    88","88    88"," 888888 "
260 DATA" 888888 ","88    88","88    88","88888888"," 888888 "
265 DATA" 999999 ","99999999","99    99","99    99","99999999"
270 DATA" 9999999","      99","99    99","99999999"," 999999 "
272 '---X--------X-X--------X-X--------X-X--------X-X--------X
275 FOR I=0 TO 11:READ L(I):NEXT I
280 FOR I=0TO 9:FOR J=0TO 9:READ X$(I,J):NEXT J:NEXT I
285 '
290 GOTO 440
295 FOR M=0TO 11
298 PRINT".";
300 W=0:DT=1
305 YR(M,D,W)=DT:DT=DT+1:D=D+1
310 IF D>6 THEN D=0:W=W+1:IF W>4 THEN W=0
315 IF DT<L(M)+1 THEN 305
320 NEXT M:SOUND1,1
325 RETURN
330 Q$=RIGHT$(STR$(Y),4):I1=VAL(MID$(Q$,1,1))
335 I2=VAL(MID$(Q$,2,1)):I3=VAL(MID$(Q$,3,1))
340 I4=VAL(MID$(Q$,4,1))
345 LPRINT:LPRINT
348 LPRINT TAB(25);"     ";VZ$;VZ$:LPRINT
350 FOR I=0TO 9
352 '--X---------------X---------X----X--------X
355 B$="                "+X$(I1,I)+"    "+X$(I2,I)
360 B$=B$+"    "+X$(I3,I)+"    "+X$(I4,I)
365 LPRINT TAB(25);B$
370 NEXT I
375 LPRINT:LPRINT TAB(25);"     ";VZ$;VZ$:LPRINT CHR$(18)
380 RETURN
```

```
385 LPRINT A$
390 FOR W=0TO 4:B$=""
395 FOR I=0TO 2:B$=B$+"    "
400 FOR J=0TO 6
405 IF YR(M+I,J,W)=0 THEN C$="   " ELSE C$=STR$(YR(M+I,J,W))
410 IF LEN(C$)<3 THEN C$=" "+C$
415 B$=B$+C$
420 NEXT J:NEXT I
425 LPRINT B$
430 NEXT W
435 LPRINT:RETURN
440 CLS:PRINTTAB(6)" VZ 200 CALENDAR":PRINT
445 PRINT" THIS PROGRAM WILL GENERATE A  CALENDAR FOR ANY ";
450 PRINT"YEAR IN THE    RANGE 1901 - 1999. ALL YOU HAVE TO ";
455 PRINT"DO IS SPECIFY THE YEAR!"
460 A$="    SU MO TU WE TH FR SA":A$=A$+A$+A$
462 '---X-----------X-X-------X-X------------------X----------X
465 H1$="           "+"JANUARY"+"                  "+"FEBRUARY"
470 H1$=H1$+"                  "+"MARCH"
475 H2$="            "+"APRIL"+"                     "
480 H2$=H2$+"MAY"+"                      "+"JUNE"
485 H3$="             "+"JULY"+"                    "+"AUGUST"
490 H3$=H3$+"                 "+"SEPTEMBER"
495 H4$="           "+"OCTOBER"+"                  "+"NOVEMBER"
500 H4$=H4$+"                 "+"DECEMBER"
502 '---X-----------X-X-------X-X------------------X----------X
504 SOUND1,1
505 INPUT" FOR WHICH YEAR WOULD YOU LIKE   A CALENDAR ";Y
510 PRINT" PLEASE WAIT -- I'M INITIALISING MY DATA....."
515 IF Y<1901 OR Y>1999 THEN PRINT:PRINT"OUT OF RANGE":GOTO 505
520 IF 4*INT(Y/4)=Y THEN L(1)=29
525 I=Y-1901:J=INT(I/4):I=I-4*J+2
530 K=5*(J-7*INT(J/7))+I:D=K-7*INT(K/7)
535 GOSUB 295
538 SOUND 3,3:PRINT:INPUT"PRESS <RETURN> WHEN READY...";I
540 PRINT" PLEASE WAIT ...I WILL NOW"
545 PRINT" PRINT-OUT YOUR CALENDAR FOR"
550 PRINT" THE YEAR-:";Y
552 GOSUB 820'TO SNOOPY ROUTINE
555 GOSUB 330
558 LPRINT CHR$(27)+"2"
560 M=0:LPRINT H1$:GOSUB 385
565 M=3:LPRINT H2$:GOSUB 385
570 M=6:LPRINT H3$:GOSUB 385
575 M=9:LPRINT H4$:GOSUB 385
580 CLS:LPRINT
585 INPUT"ANOTHER YEAR ";YY$:IF YY$="Y" THEN RUN ELSE 590
588 END
590 LPRINT CHR$(27);"@"
600 CLEAR 50:CLS:PRINT"BYE-BYE":FOR LL=1TO 10:LPRINT:NEXT LL
810 END
820 LPRINT CHR$(27);"3";CHR$(18); ' LINES 18/216 APART
830 LPRINT CHR$(15); 'CONDENSED
840 LPRINT:RETURN
1500 ERA"SNOOPYCL"
1600 SAVE"SNOOPYCL":CLS:DIR
```

Although the VZ has been around for several years, the better type utilities have been few and far between. The VZ tape Word Processor falls in this category, but unfortunately cannot be used with Disk. That used to be the case till Dave Mitchell of Rockhampton, Queensland did something about it. The result is that with D. Mitchell's two utilities you can convert your Tape W. Processor for full Disk use or Tape LOAD, Disk SAVE.

Utility one :- This will convert your Tape W. Processor for full DISK use. It will SAVE, LOAD and give DIRectories of your Disks. Needless to say, it loads/saves much faster than tape and you can see what files you have on Disk. The amount of free Disk space is also shown. One other built in feature is the ability to protect sensitive files from unautorised perusal by making them invisible to the DIRectory command.

Utility two :- This simply converts your Tape W. Processor for Tape LOAD, Disk SAVE. IT allows you to transfer your Tape W. Processor files to Disk to be used with previous converted full Disk version of W. Processor.

Both versions require a VZ 200/300 with 16K Ram Expansion.

Fortunately for H.V. VZ Users' Group D. Mitchell has donated the above conversion utilities as a fund raiser for H.V. VZ Users' Group. The club owes him its' gratitude. Thanks Dave.

The two utilities are available from H.V. VZ Users' Group for $10.00 and includes Disk, Conversion Programs, full instructions and Post and Packing anywhere in Australia.

One thing we would like to make absolutely clear is that we will not supply under any circumstances D. Smiths Tape W. Processor. You have to purchase it from D. Smith in the usual way. All we are selling is a means to convert your Tape W. Processor for Disk use.

MERGING FILES (TAPE OR DISK) :-

Although mentioned a couple times in the W. Processor instruction book about MERGING files it does not tell you how to accomplish it. The procedure is quite straightforward.

Step 1 :- Before MERGING files make sure you have both files saved to tape or disk. It is very easy to lose or corrupt one of the files.

Step 2 :- Load first file from tape or disk. Enter EDIT mode and move the cursor to bottom of text. Press RETURN about 4 times. Return to main menu.

Step 3 :- Load second file from tape or disk and the two files will be MERGED. The second file will be appended to the bottom of file one.

Any number of files can be MERGED up to W.P. memory capacity. Failure to move cursor to bottom of text will wipe out previous text from cursor position down. Use edit commands to rearange or edit your MERGED files. Like allways, test out the procedure till you are satisfied you understand its operation.

# INTRODUCTION TO PROGRAMMING 

Part 2 - by Bob Kitch.

As mentioned in Part 1 of this series, the programming task is a large and complex feat of organization and requires a wide range of skills. It is possible, and best, to break the task down into six segments - each of which must be thought about, planned and then carried out to ensure the successful completion of a software project. Even a small program requires that a cursory consideration of the six segments be made - although some of them may be quickly passed over as trivial. But it is certain that larger programs (more than 200 lines) require careful planning for success.

Before describing the six steps, it is worth thinking about "What makes a GOOD program?"

A program may be judged from a number of different standpoints; each is not necessarily mutually exclusive and sometimes some conflicts require that a trade-off be made.

The first criteria is that a program should be EFFICIENT. Efficiency can be considered from a number of varying view points. For example, optimization of the run-time can be considered as efficient. Also, reduction in storage requirements for both program code and variables can be considered as efficient programming. Furthermore, and particularly if one is developing software commercially, then efficiency can be measured in terms of the actual time required to get an applications program running and the ease of maintenance of that code. The use of appropriate data types and data structures can greatly improve the efficiency of a program. The selection of a suitable algorithm can also assist. Finally, ease of debugging so that the program can be updated or modified may be considered desirable.

The second criteria is GENERALITY and it is here perhaps that so many programs "score" so poorly. Rather than a program being written to solve a particular chore, it should be broadly written to handle a wide range of problems. The use of subroutines and functions developed and debugged previously can enormously improve programming productivity. Often a simple substitution of a variable for a constant in a program can broaden the the applicability of the program significantly.

The final criteria is ELEGANCE, which is a little harder to both define and achieve. An elegant program is one that is simple and ingenious, and possibly uses an algorithm or data structure that may not be immediately obvious to the application. The so-called "programmer's tricks" are often elegant solutions to a programming problem; but beware, some are attempts by programmers to conceal their programming stategy.

These then, are general guidelines to try and attain in your programming and by which to judge a particular programming effort as good, mediocre or poor. Notice that they are not language specific comments and are equally applicable to any programming language or exercise.

To return to the six steps in the programming task - I will briefly discuss each in turn and ask that you consider each one when embarking upon your next programming exercise. Also as one proceeds through the steps, it is often necessary to recycle back through some of the preceeding steps, to iteratively improve the exercise and your understanding of ideas.

1. PROJECT SELECTION. This may appear trivial, but we all have too many ideas for programs and rarely know which one to tackle next. Also be honest with yourself; some of the projects are probably too ambitious for your existing skills and an attempt upon these will possibly result in frustration and perhaps failure. Choose an exercise that is challenging and worthwhile. Try not to "reinvent the wheel", try to be aware through reading magazines or discussing with other Users what programs are already available. Modifying an existing program to suit your specifications is sometimes quicker - it also allows you to study how other programmers tackle problems. O.K., so now you have an idea or problem that you wish to tackle and solve.

2. PROJECT FEASIBILITY. Again be honest. Do you have the hardware, software and know-how to achieve the result? Its is not really much use trying to write large business-oriented data base programs for an 8K tape-based VZ! Check that the task is reasonable.

3. PROJECT DEFINITION. This is where the idea starts to get translated into a reality. It is also the phase where generality can be written in. It is easiest to start by thinking about the input to the program. Is it keyboard oriented, or is it to come from a programmble I/O port? Perhaps the program reads only DATA statments to configure itself or maybe the program must check if a printer is connected to the sytem? Start defining what the input will look like. Assign variable names with meaningful mnemonic names at this stage also.

Next, define the output expected from the program. Is it to write to tape and in what format? Perhaps it is to be screen oriented - can sound be used - or perhaps voice synthesis to tell the operator what is going on? Plan very carefully and fully the layout of the expected output as this is how Users will initially perceive the quality of the program.

After defining the I/O for the program we should now have a feel for the anticipated range of parameters that the program is meant to accept and also handle. This brings in the very important concept of defining the BOUNDS within which the program must function correctly. Following on from this, is range checking of all input parameters so that the program cannot go beyond the range that it was designed for and give unexpected results. A number of warning messages must be built into the program along with error capture and recovery routines. It is failure to define the operating bounds of a program that causes most crashes or rogue behaviour. Even the definition of integer variables at this stage can assist by improving program execution time and reducing storage requirements.

The definition stage should be roughed out on pieces of paper kept for later reference. Perhaps better, is to use an old exercise book. Another benefit of this is that over a period of months your progress can be measured and your growth of programming ideas recorded. Another benefit (although I hardly dare mention it!) is that if, after the coding stage, a system crash occurs and you didn't SAVE the program, then all is not lost - at least an outline of the program remains.

4. DESIGN PHASE. Having sorted out I/O and operating bounds, the actual selection of an algorithm to achieve the result is commenced. By this time some idea of the number of variables required and their type should have begun to gel. This is also the stage where your basic honesty in stages 1 and 2 may catch up with you! Data structure organisation and algorithm selection are really experience-related skills.

Hence the suggestion to read and/or modify existing programs. But do not despair - practice makes perfect.

5. IMPLEMENTATION PHASE. To date very little actual coding should have been done; in fact the computer need not even have been turned on! Some people may be surprised at how late in the task the computer actually enters into the picture. An awful lot of planning and organizing can be done off the computer and on the "backs of old envelopes".

It is also at this stage that the choice of programming language should be made. Is the program time dependant? If it is, then it should probably be written is Assembler. If the actual timing is not so critical then writing in BASIC with its diagnostics and helpful features (so typical of a high level language) deem it sensible. Experienced programmers will probably use a bit of each in practice. A very sensible compromise is to develop the program in interpreted BASIC and once finalized and debugged, compile the BASIC code to speed up execution.

6. EVALUATION PHASE. This is the moment of truth! Does the program fulfill all the criteria set out in the definition phase. If so, then you have succesfully achieved your task. Is the output as you expected it? Are the results correct? It is a good idea to have a standard set of data to exercise the program so that it can be quickly verified after a program alteration. Ensure that all logical paths through the program have been exercised so that no spurious errors of logic remain undetected. Finally, deliberately try values that are out of the intended bounds of the program to ensure that you have trapped them and that the program recovers from this type of misuse above and beyond its' intended design range.

As mentioned in part 1, the offer on programming queries still stands. If writing to Bob Kitch please include a SAE.

BOB KITCH (07) 378 3745
7 EURELLA STREET
KENMORE 4069
QUEENSLAND

## OTHER VZ PUBLICATIONS . . . . .

VZ USER
MARK HARWOOD
P.O. BOX 154
DURAL 2158
N.S.W.

$15.00 PER ANUM

LE'VZ 200/300 OOP
J.C.E. D'ALTON
39 AGNES STreet
TOOWONG 4066
QUEENSLAND

$1.00 PER ISSUE

WRITE PROTECT OVERRIDE SWITCH :- By Joe Leon

This is to my knowledge the first hardware modification for the VZ Disk Drive. As most of you may be aware it is possible to use the reverse side of your Disks for program storage simply by cutting a matching notch on the opposite side of the Disk. Commercial Notchers are available for that purpose or you can use scissors if care is taken. And now to the circuit.

Only the circuit within the box is needed. The rest is part of the disk drive circuitry and is shown so we can understand what is happening. When a notched disk is inserted in the drive a small light shines through the notch on the disk and illuminates the photo transistor (STR1). This has the effect of grounding pin 13 of U3. When that happens the disk can be written to. If you have a Write Protect Tag covering the notch then a 'DISK WRITE PROTECTED' message apears if you try to write to the disk.

It follows then that to overcome the write protect function we must ground pin 13 of U3. The easiest way to ground pin 13 of U3 is to connect a wire to the 4K7 resistor and the other end to the switch as shown in the diagram. I used a Flashing Red Led as a warning for which no resistor is required if used with 5 Volts.

If you have trouble getting a flashing red led from D. Smiths try TANDY'S. This is a simple project that should'nt present any problems. Only the top cover of the disk drive need be removed and just follow the diagrams. Once installed the switch will allow you to write to the flip side of disks without notching or over a write protect tag on the front side. Exercise care when using the WRITE PROTECT OVERRIDE SWITCH as your disks are no longer WRITE PROTECTED.

While you have the cover off there are two more modifications you can try. When the VZ is accessing the drive you only get a dull red glow from the motor on led. It can be easily replaced by a super bright 5mm LED. The easiest way to do it without removing the circuit board is to cut the pins of the old led as close to the body of the led as you can. Then place the super bright led over the old pins and push it towards the front till it nearly touches the rectangular lens and solder in place. Use caution and a heatsink when soldering the led as they are heat sensitive. Make sure the led goes in the right way. The negative side usually has a flat side on the body. See circuit diagram.

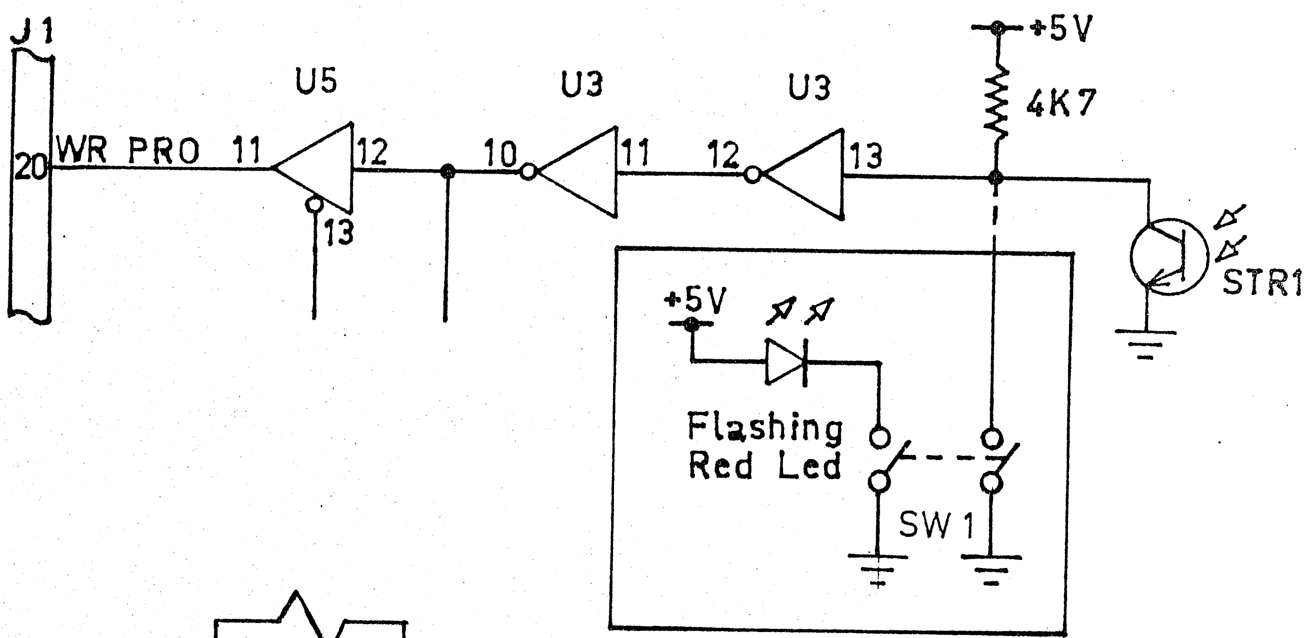
POWER ON/OFF SWITCH :-

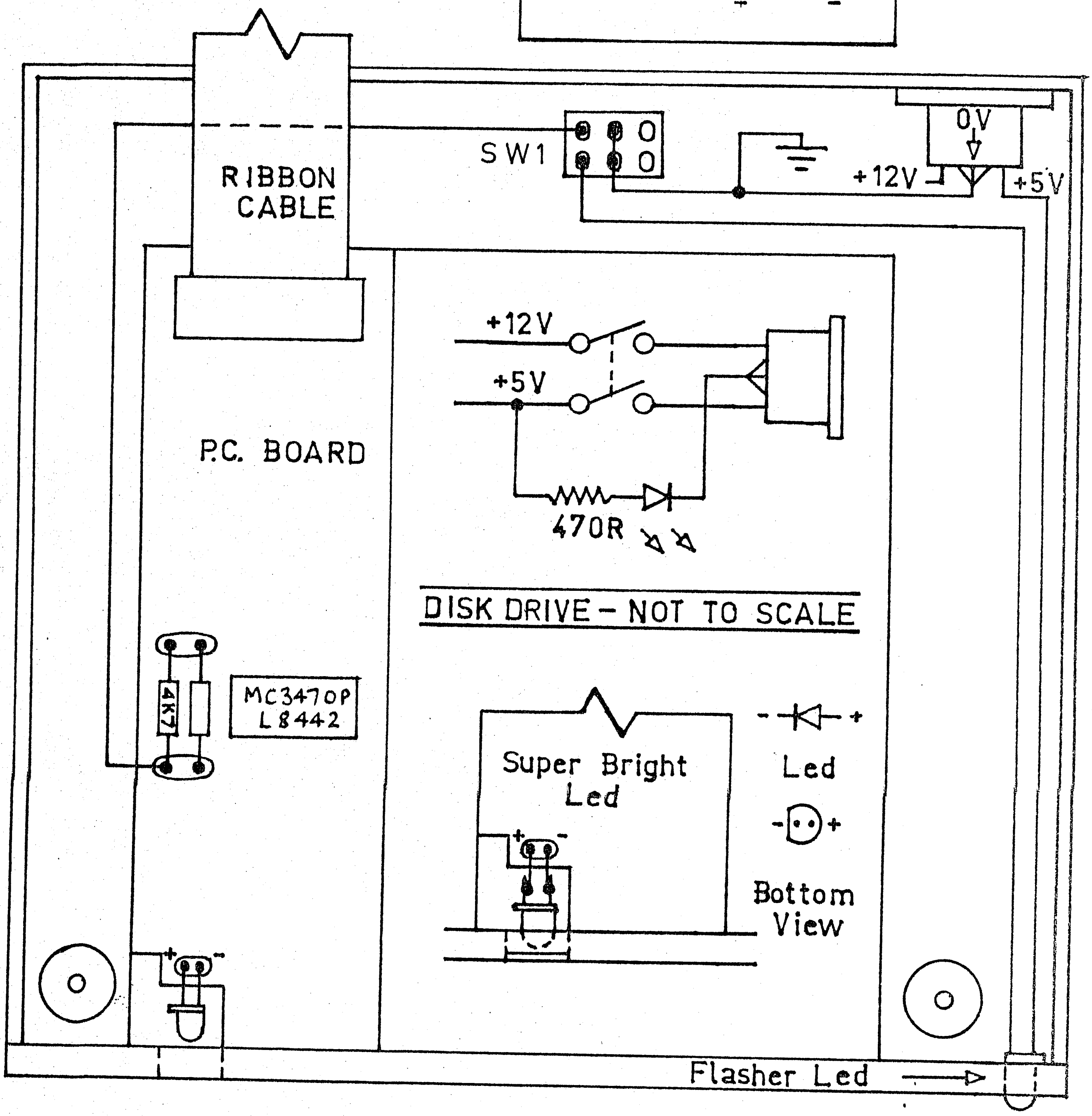
Again this is a simple project. Just a switch and a led with dropping resistor is required. Or you can dispense with the switch and just use the led with its resistor. Again it's up to you where the components are mounted.

If your drive is still under warranty then any modifications will void it. Dont forget the responsibility is yours for any mishaps. If you doubt your ability to carry out the modifications then dont.

WRITE PROTECT OVERRIDE SWITCH :-

By Joe Leon



SO YOU PRESSED NEW INSTEAD OF RUN

DON'T WORRY :- When you use the NEW command the VZ POKES TWO ZERO BYTES into the start of the basic program to replace these bytes is very easy What they tell the computer is the address of the next line of the basic program.

TRY THIS :- Type in line 10 then return
Type in line 20 then return

```
10 CLS
20 PRINT
```

WITH NO LINE NUMBER TYPE & press return

```
FOR I = 31465 to 31476 : PRINT I ; PEEK( i ): NEXT
```

LISTED ON THE VIDEO SHOULD BE

```
31465  239  ; ADDRESS OF THE NEXT LINE (LSB)
31466  122  ; ADDRESS OF THE NEXT LINE (MSB)
31467   10  ; LINE NUMBER (LSB)
31468    0  ; LINE NUMBER (MSB)
31469  132  ; TOKEN FOR CLS
31470    0  ; END OF LINE
31471  245  ; ADDRESS OF NEXT LINE (LSB)
31472  122  ; ADDRESS OF NEXT LINE (MSB)
31473   20  ; LINE NUMBER (LSB)
31474    0  ; LINE NUMBER (MSB)
31475  178  ; TOKEN FOR PRINT
31476    0  ; END OF LINE
```

31465 & 31466 contains the address of the next line. WITH simple maths it is easy to find this address :-

$$239 + 256 * 122 = 31471$$

CHECK 31471 YOU WILL SEE IT IS THE ADDRESS OF THE NEXT Line.

List the program you typed. Now press NEW & return.
List again, WHAT DID YOU GET ? WAS IT NOTHING ?

TYPE :- PRINT PEEK( 31465 ) ; PEEK( 31466 ) & press return.
IT SHOULD BE --- 0 0

TYPE :- POKE 31465,239 : POKE 31466,122 : LIST & press return

You have just got the program back from the UNKNOWN !

Here is an example in basic of what the machine code does.

```
10 A = 31467
20 B = PEEK( A )
30 IF B <> 0 THEN A=A+1 : GOTO 20
40 IF B = 0 THEN A=A+1
50 C=A-(256*INT(A/256)
60 D=INT(A/256)
70 POKE 31465,C
80 POKE 31466,D
```

In line 10 'A' equals the normal start of the basic program plus two.
Line 20 'B' looks at what is at location 'A'
Line 30 checks if 'B' is bigger or smaller than zero and adds one to 'A' then goes to line 20
Line 40 checks if 'B' equals zero then 'A' is added with one.(the zero inicates the end of the basic line.)
Line 50 works out the LEAST SIGNIFICIANT BYTE (LSB).
Line 60 works out the MOST SIGNIFICIANT BYTE.
Line 70 POKES 31465 with the LSB.
Line 80 POKES 31466 with the MSB.

The above program is only an example to help you understand the assembler code later.

FOR THOSE WITHOUT AN ASSEMBLER PROGRAM.

```
10 FOR I = -20480 TO -20391
20 READ A:POKE I,A: NEXT
30DATA205,201,1,33,39,176,205,167,40,17,236,122,19,26,254,0,32
40DATA250,19,235,34,233,122,33,233,122,34,164,120,205,248,26,35
50DATA34,249,120,195,25,26,13,32,32,32,32,32,32,32,32,32,32,32
60DATA82,69,83,84,79,82,69,13,13,32,32,32,32,32,66,89,32,68,46
70DATA77,73,84,67,72,69,76,76,32,32,86,75,52,75,68,65,0,0,0,0
80 CLS:PRINT"SAVE TO DISK OR TAPE   (D/T)"
90 A1$=INKEY$:A$=INKEY$:IFA$<>"D"ANDA$<>"T"THEN90
100 SOUND30,1:IFA$="T"THEN180
110 IF PEEK( 16384 ) = 170,140
120 PRINT"NO DISK DRIVE YOU NAUGHTY PERSON"
130 GOTO 180
140 PRINT"INSERT DISK,CLOSE DOOR & PRESS  RETURN"
150 IFINKEY$<>CHR$(13),150
155 SOUND30,1
160 BSAVE"RESTORE",B000,B059
170 END
180 FOR I = -24576 TO -24526
190 READ A: POKE I,A: NEXT
200 PRINT"INSERT CASSETTE,PRESS PLAY &    RECORD THEN RETURN"
210 IF INKEY$ <> CHR$(13),210
220 SOUND30,1
230 POKE 30862,0:POKE 30863,160:X=USR(0)
240 END
250 DATA33,0,176,34,164,120,33,89,176,34,249,120
260 DATA33,38,160,14,241,243,205,172,52,251,33,233,122
270 DATA34,164,120,205,248,26,35,34,249,120
280 DATA195,25,26,34,82,69,83,84,79,82,69,34,0,0,0,0
```

Lines 10 to 70 contains the data for restore.
Lines 80 to 100 checks for saving to disk or tape.
Line 110 checks for a disk drive.
Lines 120 to 130 prints an error then goes to tape save.
Lines 140 to 170 saves to disk.
Lines 180 to 190 sets up data for AUTO-RUN tape save.
Lines 200 to 210 prints message & waits for the return key to be pressed.
Line 230 jumps to the machine code for AUTO-RUN TAPE SAVE.
Lines 250 to 280 data for auto-run tape save.

FOR MULTIPLE COPIES :- DISK TYPE GOTO 140
TAPE TYPE GOTO 200.

THE ASSEMBLER CODE.

```
      CALL 01C9H     ;clear screen.
      LD HL,MES      ;load HL with the message address.
      CALL 28A7H     ;print the message.
      LD DE,7AECH    ;load de with 7AEC hex.
SCAN  INC DE         ;add one to de.
      LD A,(DE)      ;load 'A' with what is at the address of DE.
      CP 00H         ;test - is it a zero.
      JR NZ,SCAN     ;NO - go back to scan & do it again.
                     ;YES - continue.
      INC DE         ;add one to DE.
      EX DE,HL       ;exchange regesters.
      LD (7AE9H),HL;load the contents of HL into 7AE9 HEX.
      LD HL,7AE9H    ;load HL with 7AE9 HEX.
      LD (78A4H),HL;load start of program with HL.
      CALL 1AF8H     ;reset program statement table.
      INC HL         ;add one to HL.
      LD (78F9H),HL;load end of program with HL.
      JP 1A19H       ;JUMP TO BASIC.
MES   EQU $          ;start of message.
      DEFB ODH       ;carriage return.
* [11 spaces] RESTORE*
      DEFB ODH       ;carriage return.
      DEFB ODH       ;carriage return.
* [5 spaces] BY D.MITCHELL*
* [2 spaces] VK4KDA*
      NOP            ;end of message.
```

```
ORGIN              0B00 HEX.
START OF PROGRAM   B000 HEX.
END OF PROGRAM     B059 HEX.
```

NOTE :- PROGRAM WILL RUN IN A STANDARD VZ300 OR VZ200 + 16K.

## HIGH GAMES SCORE - - - - - - - -

| | | |
|---|---|---|
| ASTEROIDS | 35020 | MATTHEW TAYLOR |
| CIRCUS | 1350 | GEOFFREY KEEN |
| DAWN PATROL | 60200 | MATTHEW TAYLOR |
| DIG OUT | 83700 | GEOFFREY KEEN |
| GALAXON | 29200 | MATTHEW TAYLOR |
| GHOST HUNTERS | 18780 | CHRISTIAN WARNER |
| HAMBURGER SAM | 83600 | GEOFFREY KEEN |
| HOPPY | 10740 | MATTHEW TAYLOR |
| LADDER CHALLENGE | 25400 | MATTHEW TAYLOR |
| PANIK | 16720 | BARRY KEEN |
| PLANET PATROL | 1591 | WARREN KEEN |
| ROAD WARRIOR | 28370 | MATTHEW TAYLOR |
| SPACE INVADERS | 17290 | MATTHEW TAYLOR |
| STAR BLASTER | 812 | ADAM MAGEE |

SHIFT LOCK SWITCHES :- By Joe Leon

Instead of using a mechanical push on/push off switch an electronic FLIP FLOP can used. Two versions are shown. I'll leave it up to you which version you use.

CIRCUIT 1 :-

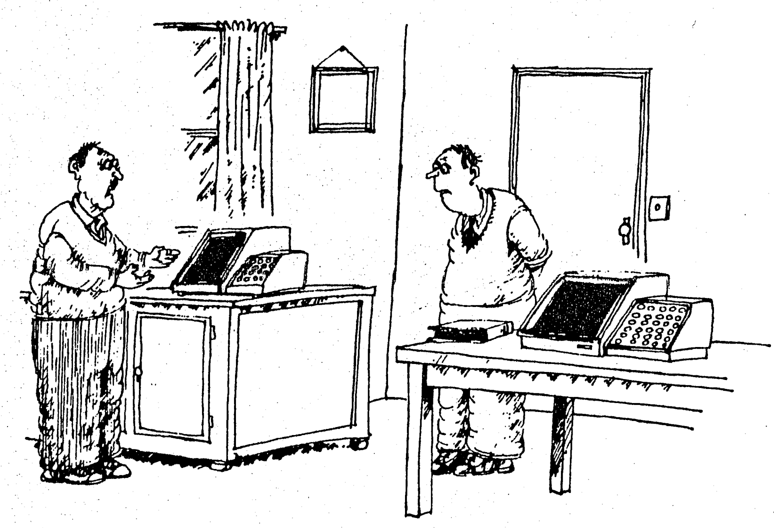
This circuit uses one half of a dual flip flop. This is the one I used. The P.B. SW. (Push Button Switch) shown can be a small momentary switch mounted anywhere convenient. In my case I used the RIGHT SHIFT KEY on the VZ 300 Keyboard. If you decide to use this option then the track on either side of the Right Shift Key on the Keyboard P.C.B. must be cut. Solder a thin wire to each side of the KEYPAD on the track leading to the isolated Shift lock switch. The other ends of the two wires go as shown on the circuit diagram. While you have the Keyboard apart you may decide to mount the 3mm LED as I did. It went in the Right Shift Key Cap itself, right above the "I" in the word SHIFT. It looks neat. The choice is yours where it goes. Connecting the Transistor to the keyboard matrix is straightforward. When built the LED will let you know when the switch is on. If the LED is on, but Shift Lock is not then try reversing the leads from the Transistor.

CIRCUIT 2 :-

This circuit does the same job as above and is intended for Persons who built the softstart switch as shown in the November issue, page 8. It uses the left over gates from that circuit.

CAUTION :-

Both IC'S are CMOS type and special handling precautions must be observed. Do not touch the pins on the IC'S as static can destroy them. Use a socket for the IC just to be safe. Also all unused INPUTS must be grounded or taken to +5V. Pinouts for both IC'S are shown to help intending constructors.

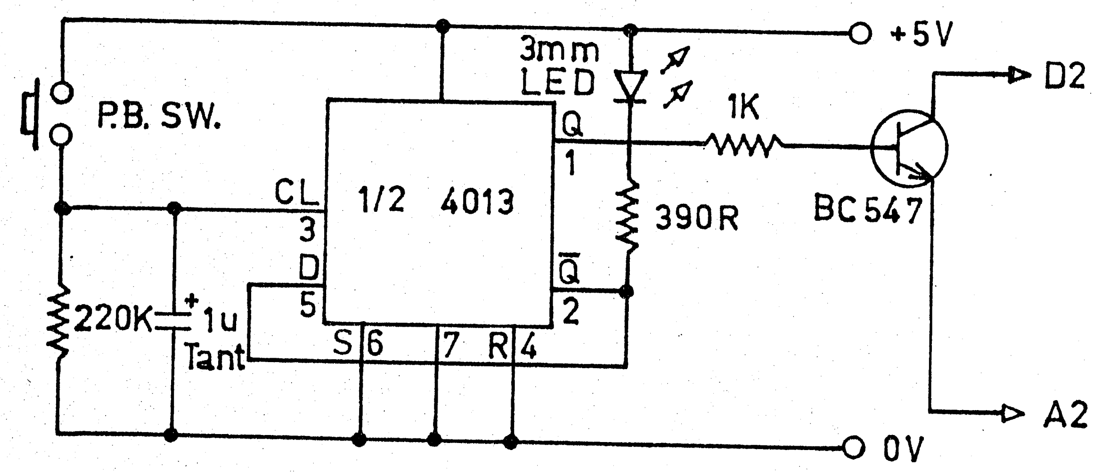


*'And I bought this one to explain the manual of the first one.'*

SHIFT LOCK SWITCHES :-

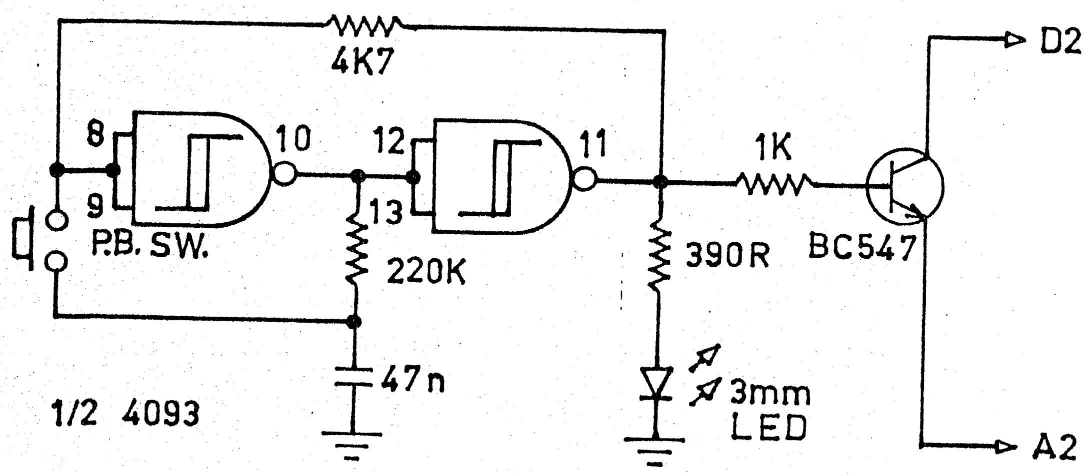
By Joe LEon

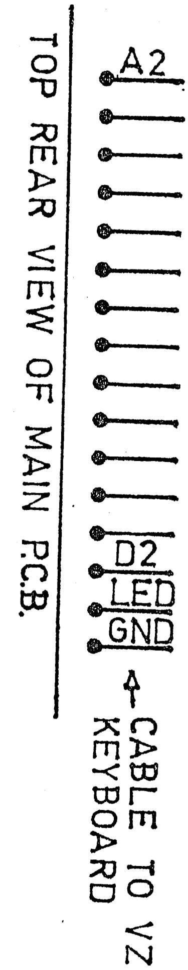
## CIRCUIT 1



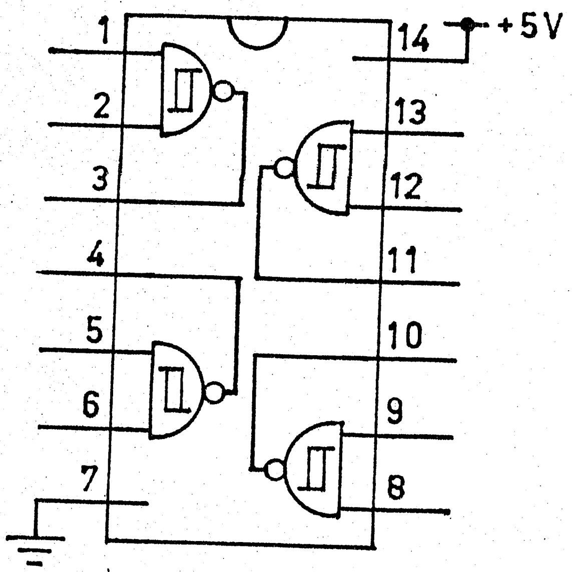

GROUND PINS 8,9,10,11.

## CIRCUIT 2

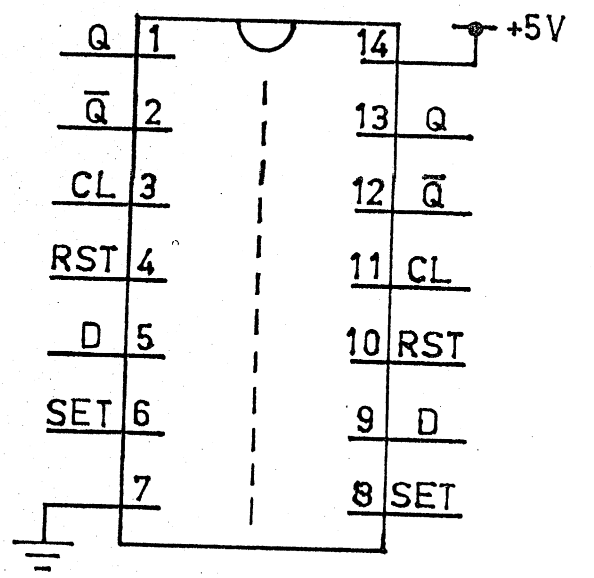



4093




4013



NOTE :- See issue 6, page 18 for LLISTing of DLINE.

DLINE is the deluxe version of LINE. When you have entered DLINE, check the accuracy of the DATA statements with this direct command:

```
FORR=29184 TO 29260:READA:Z=Z+A:NEXT:PRINTZ
```

If your answer is not 6319, then recheck all the numbers in the DATA statements at the end of DLINE. When RUN5000, DLINE will ask for a line number. Enter the number of the program line you want analysed, or enter a basic word. If you enter a word, DLINE will search through the word table (token list table) for the word and print the token code for that word, then ask for a line number again.

When a line number has been entered, DLINE will ask for a token. If you want DLINE to seek out a particular token code in the line, then enter that token and DLINE will display each byte in the line in continuous sequence, pausing only when it encounters an instance of the token. If you do not want DLINE to search for a particular token, then simply press <RETURN> and DLINE will display the first five bytes of the line and pause. Press <SPACE> to single-step through the remainder of the line. DLINE will display the address of the memory cell for each byte byte in the line (first,second,third,etc.), the byte itself, then the character or basic word designated by the byte (under the ? heading) and, finally, the negative PEEK/POKE address of the byte if the address is greater than 32767. Whenever DLINE pauses, you can press <O> or <L> or <X> key to activate a particular POKE facility.

<O>: DLINE will ask for a POKE address and then POKE the number 140 (token code for LET) to that address.

<L>: DLINE will ask for a POKE address, then for a TOKEN, and POKE the token number to the address.

<X>: DLINE will ask for a TOKEN and then POKE the token number to the last address displayed on the screen.

<6>: Pressing this key will reRUN DLINE.

```
50300 IFZ=192THEN PRINT"VARPTR";:RETURN
```

This line in DLINE recognises the token for VARPTR and lists this word when it encounters the token. Other similar lines can be inserted into DLINE for other words that are not functional in the word table. They should have line numbers greater than 50300 and less than 50320.

```
EG. 50302IFZ=196THEN PRINT"STRING$";:RETURN
```

Such lines should be inserted after DLINE has been merged with another program, because DLINE is already close to the maximum length of a program that can be merged using the MERGE routine accompanying this article. Any further lines will take it over the limit. Those of you who have a disk drive can run DLINE and then BSAVE a binary program of the machine code routines used by DLINE:-

BSAVE"LCODE",7200,724C and then delete all the DATA statements and the READ/POKE routine from DLINE. The memory gained can then be used to install a number of statements for recognising and listing various nonfunctional words as indicated above.

Line 50010 should now read:

50010 IFPEEK(29223)=33ANDPEEK(29224)= 233THEN50030 and a line 50020 entered thus:

50020 INPUTB:BLOAD"LCODE"

The INPUT statement is simply a precaution to prevent you from initiating BLOADing of LCODE without a disk in the drive or the door closed (you've had that experience too have you?).

Incidentally, if you want to use VZ disk commands in conditional statements, the trick is to insert a colon between THEN and the disk command, like this:-

80 IFK=9THEN:BLOAD"PROGRAM"

The deluxe version of line is obviously too long to type in every time you want to append it to a program. Salvation is at hand in the form of a merge routine. You need only type DLINE in once, save it on tape or disk, and thenceforth use the MER program to join DLINE to other programs.

MER allows you to join one program onto the end of another so that they become a single program. Type in MER and save. Check the length of MER with this length command:-

PRINTPEEK(30969)+PEEK(30970)*25 6-31465

The answer should be 370. Then check the machine code of the MERGE routine with this command sequence:

FORR=31273TO31354:READA:B=B+A:N EXT:PRINTB

The answer should be 9118. If not, check out all the numbers in the DATA statements. When RUN, MER sets up a machine code MERGE routine in the communications region. MERGE is the program which does the adding on.

PROG 1 is the program you wish to add on.
PROG 2 is the program to which you want to add PROG 1.

The maximum length for PROG 1 is 1534 bytes. If you have any doubts as to whether PROG 1 exceeds the maximum length, then check the length with the above command. For trouble-free merging, ensure that the line numbers for all the lines in PROG 1 are greater than all the line numbers in PROG 2.

Enter the following sequence of commands to carry out a merge (CRUN and CLOAD if you are using a cassette recorder):

```
RUN"MER"      'Transfers M. Code MERGE to communications region
LOAD"PROG 1"  'Use your name for PROG 1
PRINTUSR(8)   'MERGE copies PROG 1 to video memory
LOAD"PROG 2"  'Use your name for PROG 2
PRINTUSR(8)   'MERGE adds PROG 1 onto end of PROG 2
```

Most routines you would want to merge with another program would be less than the maximum length permitted (1534).

However, should you want to add on a longer routine, you can do it in stages. When you load PROG 1, chop PROG 1 in half by using the DELETE command (see Table X) to remove the top end of PROG 1, and complete the merge. Save the result; this will be your new PROG 2. Now repeat the merge sequence of commands from LOAD"PROG 1" onwards.

This time DELETE the bottom part of PROG 1 and merge the remainder with the new PROG 2.

Once MER has been run the MERGE routine can be used to merge any number of pairs of programs without having to reRUN the MER program each time.

Merge will add on PROG 1 to PROG 2 regardless of the line numbers of the two programs. The result could be that a routine is joined to the end of PROG 2 that has the same line numbers as PROG 2 or some part of PROG 2. This is an opportunity to get two or more programs to reside in program memory simultaneously. With the proper POKEs to start of program pointer you can then RUN the program of your choice.

LLISTing for MERge :-

```
10 FORR=31273TO31354:READA:POKER,A:NEXT
20 POKE30862,41:POKE30863,122
30 DATA175,42,249,120,17,233,122,237,82,68,77,237,67,0,114,33
40 DATA233,122,17,2,114,237,176,33,71,122,34,142,120,201,237
50 DATA91,249,120,27,27,107,98,237,75,0,114,9,34,249,120
60 DATA33,2,114,237,176,33,41,122,34,142,120,175,17,233,122,98
70 DATA107,190,35,32,2,190,200,35,35,35,190,32,252,35,235,115,35
80 DATA114,24,235
```



